R Workshop

St. Stephen's College, October-November 2023

Ananya Iyengar

Table of Contents

1. Session 1: Data Manipulation

2. Session 2: Graphing

3. Session 3: Readability and Good Coding Practices

Why do we need to manipulate data?

1. Data may be available in a format that is different from what we require.
2. Need to create variables that don't exist in data.
3. The information we require may be in two different data frames, formats, etc. and we want to bring it all together.
4. We want to express the same information in a different way

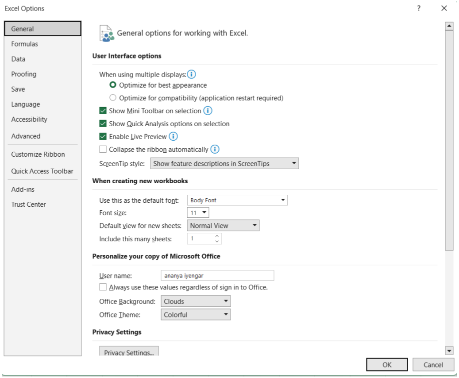
Where do we manipulate data?

We are used to working on MS Excel- it might seem simpler in some cases! So why R?

1. Proprietary VS Open Source Software
2. The GNU Project (Ihaka and Gentleman, 1997)
3. More transparent!
4. (Personal opinion) Good workflow!

But Excel is inevitable sometimes: I

A lot of data is in .xlsx or .csv format! **Solution:** Track Changes in MS Excel. how we do this!



General options for working with Excel.

User Interface options

When using multiple displays:
- ● Optimize for best appearance
- ○ Optimize for compatibility (application restart required)

☑ Show Mini Toolbar on selection

☑ Show Quick Analysis options on selection

☑ Enable Live Preview

□ Collapse the ribbon automatically

ScreenTip style: Show feature descriptions in ScreenTips

When creating new workbooks

Use this as the default font: Body Font

Font size: 11

Default view for new sheets: Normal View

Include this many sheets: 1

Personalize your copy of Microsoft Office

User name: ananya iyengar

□ Always use these values regardless of sign in to Office.

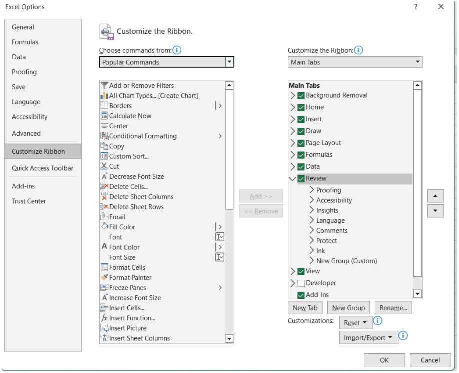
Office Background: Clouds

Office Theme: Colorful

Privacy Settings

Privacy Settings...

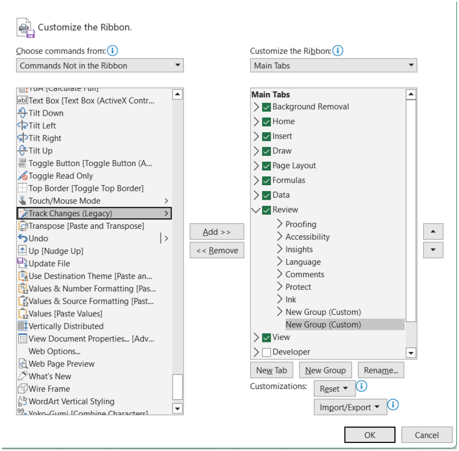
But Excel is inevitable sometimes: II



Choose commands from: Popular Commands

Customize the Ribbon: Main Tabs

But Excel is inevitable sometimes: III



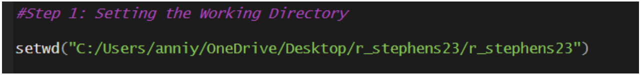
Customize the Ribbon.

Choose commands from: Commands Not in the Ribbon

Customize the Ribbon: Main Tabs

Setting up your R Workspace

1. Create a new project
2. Set your Working Directory
3. Store all your data files in your Working Directory



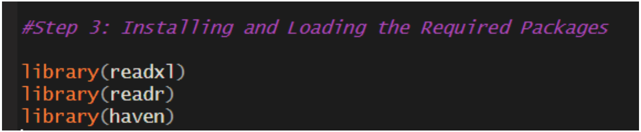
```
#Step 1: Setting the Working Directory

setwd("C:/Users/anniy/OneDrive/Desktop/r_stephens23/r_stephens23")
```

Packages to import data

1. For .xlsx or .xls files: readxl
2. For .txt files: readr
3. For .dta (STATA) files: haven

These packages are installed from the CRAN (Comprehensive R Archive Network) repository.



```
#Step 3: Installing and Loading the Required Packages

library(readxl)
library(readr)
library(haven)
```

Importing Data

Main data set: Data from Census 2011! This is data on the population and education attainment for different castes at the district level!

Other data sets for importing practice: practice.dta and pit_stop.txt!

Data Cleaning

What to look for?

1. NAs in the data: what are they? can they be resolved?
2. Are the NAs a data feature or a coding problem?
3. Is it justified to remove those observations?

These are qualitative questions are the answers depend on the context you are working on! If time permits, we will have a portion solely on imputation in the coming session!

Data Manipulation

Main package used: dplyr

Benefits: Good flow of work, easy debugging, useful functions

Function: mutate, select, filter, arrange, slice

Merging Data Sets

We typically use the merge command on base R. We can also, however, use commands such as inner_join on the *dplyr* package. There are many ways to do things on R, and we must choose methods on the basis of what suits us the best!

We covered the following:

1. Introduction to *ggplot2*: The Grammar of Graphics.
2. Making basic point graphs using the *ggplot2* package and: changing dot size, shape, opacity; log scales; setting axes label and title sizes; setting the theme of the graph; adding captions.
3. Working with different kinds of legends: labelling, colour and sizing.
4. Using the *rbind()* command.
5. Using the *facet_wrap()* command, and its importance.
6. Using the *viridis* package for better colour schemes.
7. Using the *plotly* package to create interactive and HTML exportable graphs.
8. Area plots using the *ggplot2* package.

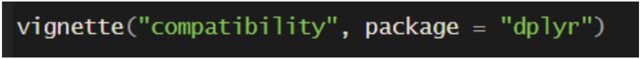
Features of Good Scientific Code

Source: Benureau and Rougier (2018). Frontiers in Neuroinformatics.

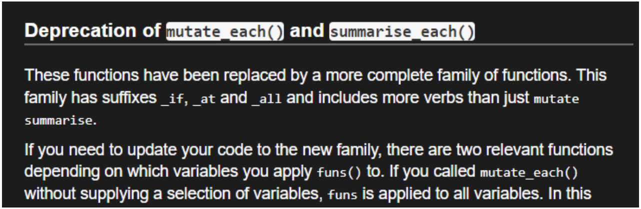
1. Re-Runnable
2. Repeatable
3. Reproducible
4. Replicable

Re-Runnable

Needs thorough code checks! An important consideration is deprecated commands.



```
vignette("compatibility", package = "dplyr")
```



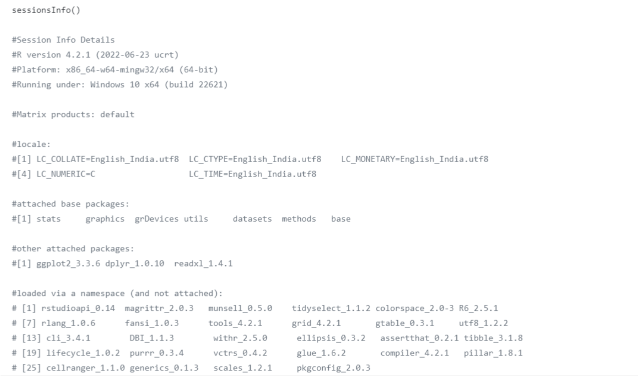
Deprecation of `mutate_each()` and `summarise_each()`

These functions have been replaced by a more complete family of functions. This family has suffixes `_if`, `_at` and `_all` and includes more verbs than just `mutate` `summarise`.

If you need to update your code to the new family, there are two relevant functions depending on which variables you apply `funs()` to. If you called `mutate_each()` without supplying a selection of variables, `funs` is applied to all variables. In this

ALWAYS mention what version of system is used by the *sessionInfo()* command.

Re-Runnable



```
sessionsInfo()

#Session Info Details
#R version 4.2.1 (2022-06-23 ucrt)
#Platform: x86_64-w64-mingw32/x64 (64-bit)
#Running under: Windows 10 x64 (build 22621)

#Matrix products: default

#locale:
#[1] LC_COLLATE=English_India.utf8  LC_CTYPE=English_India.utf8    LC_MONETARY=English_India.utf8
#[4] LC_NUMERIC=C                    LC_TIME=English_India.utf8

#attached base packages:
#[1] stats     graphics  grDevices utils     datasets  methods   base

#other attached packages:
#[1] ggplot2_3.3.6 dplyr_1.0.10  readxl_1.4.1

#loaded via a namespace (and not attached):
# [1] rstudioapi_0.14  magrittr_2.0.3   munsell_0.5.0    tidyselect_1.1.2 colorspace_2.0-3 R6_2.5.1
# [7] rlang_1.0.6      fansi_1.0.3      tools_4.2.1      grid_4.2.1       gtable_0.3.1     utf8_1.2.2
# [13] cli_3.4.1        DBI_1.1.3        withr_2.5.0      ellipsis_0.3.2   assertthat_0.2.1 tibble_3.1.8
# [19] lifecycle_1.0.2  purrr_0.3.4      vctrs_0.4.2      glue_1.6.2       compiler_4.2.1   pillar_1.8.1
# [25] cellranger_1.1.0 generics_0.1.3   scales_1.2.1     pkgconfig_2.0.3
```

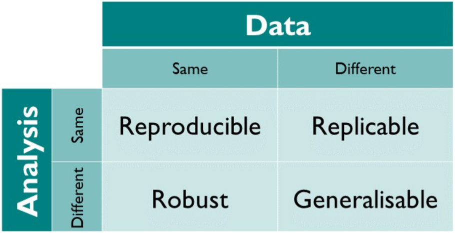
Repeatable

1. Successive runs of the code must give the same output.
2. Important when using randomisation.
3. set.seed() guarantees the same random values.

Reproducible

Original data + Code = Same Result

Source: The Turing Way

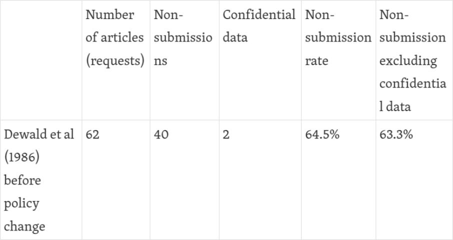


		Data	
		Same	Different
Analysis	Same	Reproducible	Replicable
	Different	Robust	Generalisable

Fig. 5 How the Turing Way defines reproducible research

Reproducible

1. Reinhart and Rogoff (2010)
2. AEA Rules about code and data availability, but varies journal to journal
3. This is not a new conversation! Dewald et a. (1980) in the American Economic Review.



	Number of articles (requests)	Non-submissions	Confidential data	Non-submission rate	Non-submission excluding confidential data
Dewald et al (1986) before policy change	62	40	2	64.5%	63.3%

Replicable

Definition

The ability of a researcher to duplicate the results of a prior study if the same procedures are followed but new data is collected.

Replication studies are important not just to validate the methods but look at the generalisability of results i.e. External Validity!

None of this is possible without open data!

Ragnar Frisch, 1993

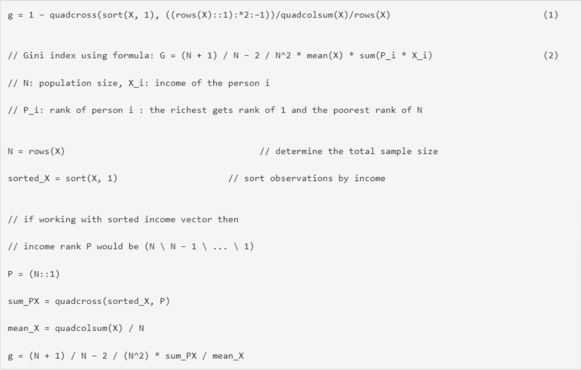
...the original data will, as a rule, be published unless their volume is excessive [...] to stimulate criticism, control and further studies

Brief Introduction to Creative Commons Licenses

1. Public copyright license
2. Share, use and build upon the work of others
3. Instituted in 2002
4. Lawrence Lessig, Eric Eldred
5. Aaron Swartz at MIT
6. Used by the AEA

How do we do all this?

1. Well-commented code. *Source:* Lokshin and Sajaia (2022) + Own Example



```
g = 1 - quadcross(sort(X, 1), ((rows(X)::1):*2:-1))/quadcolsum(X)/rows(X)                (1)


// Gini index using formula: G = (N + 1) / N - 2 / N^2 * mean(X) * sum(P_i * X_i)         (2)

// N: population size, X_i: income of the person i

// P_i: rank of person i : the richest gets rank of 1 and the poorest rank of N


N = rows(X)                                      // determine the total sample size

sorted_X = sort(X, 1)                      // sort observations by income


// if working with sorted income vector then

// income rank P would be (N \ N - 1 \ ... \ 1)

P = (N::1)

sum_PX = quadcross(sorted_X, P)

mean_X = quadcolsum(X) / N

g = (N + 1) / N - 2 / (N^2) * sum_PX / mean_X
```

README files

Source: Social Science Data Editors website

1. Contents of the data submission
2. In case of secondary data, date accessed
3. Source data for all graphs and figures
4. List of data sets
5. Computational requirements: software, memory, run-time
6. Code description
7. Other statements and declarations

Show example: The Economics of Internal Migration paper

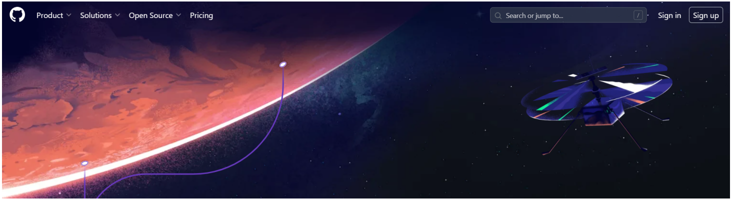
Documenting all this: Miguel and Kremer

They provide:

1. Data users guide
2. Codebook
3. Replication Manual

How do we do this? Introduction to GitHub!

GitHub is a cloud-based code hosting service to store code, data, collaborate with other researchers and for version control of software. Our use is predominantly for storing code + data for our projects.



Introduction to GitHub Web

1. Create account
2. Create your personalised README
3. Make repository to store your code, data and output
4. README and License for your repository

Introduction to GitHub Web

This is only the tip of the iceberg! There is a lot more to know about using Git (which $\neq$ GitHub). But for our purposes + time constraints, we just want to learn how to store our work (as students) in a way that is accessible, safe and transparent!